Proceedings of the 1991 IEEE International Symposium
on Intelligent Control
13 - 15 August 1991, Arlington, Virginia, U.S.A.

Calibrating a Mobile Camera's Extrinsic Parameters with Respect to Its Platform

Y.-L. Chang and J. K. Aggarwal

Computer and Vision Research Center
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78712

ABSTRACT

This paper addresses the problem of estimating the fixed rotation and translation between the camera's and the mobile robot's coordinate systems given a sequence of monocular images and robot movements. Existing hand/eye calibration algorithms are not directly applicable because they require the robot hand to have at least two rotational degrees of freedom, while a mobile robot can usually execute only planar motion. By using the proper representation for camera rotation, the proposed algorithm decomposes the calibration task, and thus is able to calibrate all the three rotational degrees of freedom and the two translational degrees of freedom. The remaining translational degree of freedom is not needed for the purpose of camera-centered robot vision applications. Furthermore, to recover the camera's rotation motion between two images, the proposed algorithm uses inverse perspective geometry constraints on a rectangular corner. Complicated calibration patterns are not needed and, thus, the proposed algorithm can be easily implemented and used in structured environments.

1. Introduction

In sensor-based robot control applications, the vision system is sometimes mounted on a robot hand (eye-in-hand) or on a mobile robot (navigation and active vision). In either case, the robot motion is initially defined with respect to the robot coordinate system, not the camera system. Therefore, before we can use any quantitative information from the vision system, we have to know the relationship between the robot and the camera coordinate systems, in addition to the camera's "intrinsic" parameters, such as focal length. The eye-in-hand calibration problem has been studied before [9, 11]. But these algorithms are not applicable to mobile robots because they require the robot hand to have at least two rotational degrees of freedom, while a mobile robot usually has only one. This paper develops a new calibration algorithm for the mobile robot case by accomplishing the following goals:

1) We show that if the robot's rotational axis is fixed during the calibration, then the unknown robot/camera transformation can still be recovered up to only one unknown translation; this unknown translation may not be needed for some robot vision applications.

2) Using the appropriate representation for the camera's rotation, we can decompose the calibration task into two stages. Through algebraic manipulation, we are able to find the close-form solutions of all the parameters.

3) From the numerical aspect, the calibration of the camera's extrinsic parameters is similar to the well-studied structure from motion problem in the sense that we also compute the rotation and translation using the 2D line and point correspondence. Thus, the computation algorithm may have similar problems with outliers [5]. We resort to the robust estimation techniques and successfully solve the problem.

4) Another goal of our work is to replace the full-scale camera calibration used in [11] with a simpler algorithm based on the inverse perspective geometry constraints for a rectangular corner [6]. Our proposed algorithm can thus use 3D structures that are abundant in indoor scenes instead of preset calibration references.

The rest of the paper is organized as follows. Section 2 defines the problem and reviews previous work. Section 3 presents our proposed algorithm. Section 4 then shows the experimental results of our algorithms. Finally, section 5 concludes the paper.

2. Previous work

Figure 1 compactly explains the hand/eye calibration problem. Henceforth, we use the following conventions. H stands for the 4 by 4 homogeneous transformation matrix, i.e., if (x, y, z) are the original coordinates and (x', y', z') are the coordinates in the new system, then

$$[x' \;\; y' \;\; z' \;\; 1]^t = H\,[x \;\; y \;\; z \;\; 1]^t, \text{ where } H = \begin{bmatrix} R & T \\ 0\,0\,0 & 1 \end{bmatrix}. \qquad (1)$$

R is the rotation matrix and T is the translation vector. Superscript ^{t} stands for the transpose. Subscript $_c$ indicates the camera system and subscript $_g$ represents the robot. Subscripts $_i$ and $_j$ represent two different poses of the robot as well as the camera. Therefore, H_{gij} denotes the transformation between the two robot coordinate systems. H_{cij} is the transformation between the two camera coordinate systems. H_{cg} is the unknown homogeneous coordinate transformation matrix between the camera and the robot coordinate systems. The hand/eye calibration problem can be simply stated as: Given H_{cij} and H_{gij}, find H_{cg} using the relation:

$$H_{cg}\, H_{cij} = H_{gij}\, H_{cg}. \qquad (2)$$

Simply replacing each H matrix by its R and T components, the calibration of H_{cg} can be decomposed into the translation and the rotation parts:

$$R_{cg}\, R_{cij} = R_{gij}\, R_{cg}, \qquad (3)$$

$$R_{cg}\, T_{cij} - T_{gij} = R_{gij}\, T_{cg} - T_{cg}. \qquad (4)$$

In [11], Tsai and Lenz first apply the full-scale camera calibration algorithm [10] to compute the camera's intrinsic and extrinsic parameters (rotation and translation) at each location with respect to a fixed calibration reference, i.e., the transformation matrices H_{ci} and H_{cj} in Figure 1. The transformation between the two camera systems can then be computed as

$$H_{cij} = H_{ci}^{-1}\, H_{cj}. \qquad (5)$$

A large part of [11] is devoted to simplifying the computation of (2) using geometrical observations and algebraic manipulations. In addition, an analytical error modeling of the algorithm is presented and verified by experimental data.

Shiu and Ahmad [9] assume that the H_{cij} is given and solve (2) by geometrical and algebraic methods. Their formulas are different from those in [11] mainly because they use different representations and intermediate variables. The authors also derive the conditions under which (2) can be solved uniquely. But error analysis is only given via a numerical simulation.

Puget and Skordas [8] use Faugeras and Toscani's method [3] to calibrate a mobile camera at several different poses. However, their goal is not to calibrate the hand/eye transformation matrix H_{cg}, but to improve the accuracies of the camera's intrinsic and extrinsic parameters computed at each location.

Zhang and Faugeras [12] calibrate the eye/wheel transformation between the mobile robot and the trinocular vision system mounted on it. They implicitly assume that the vertical axis of the trinocular system is parallel to the robot's vertical axis. Thus, they only need to recover the intersection of the robot's rotation axis and the ground plane (y = 0) of the tri-nocular vision system.

Brooks *et al.* [1] pioneer the idea of mobile camera self-calibration. However, they consider vertical image lines and forward robot motion only. They use a stereo vision system instead of a single camera. Furthermore, the design goal of their work is obstacle avoidance, not a general purpose hand/eye calibration algorithm.

The algorithm proposed in this paper begins with Eqs. (3) and (4). However, since a mobile robot usually can only rotate around its vertical axis, algorithms studied in [9] and [11] are not directly applicable. The next section derives a new algorithm to solve the problem by decomposing the camera rotation appropriately.

3. The proposed algorithm

The following work uses the pin-hole camera model and assumes that the camera's intrinsic parameters, such as focal length, are already given by an off-line calibration process [10]. It also assumes that the direction of a robot's rotational axis is stationary. The assumption means that the ground surface is smooth, while small bumps and dips are treated as random noise in a robot's motion. Note that the assumption is only needed in the calibration stage and that it is easy to satisfy for indoor environments.

Figure 1 defines the robot's and the camera's coordinate systems. The z-axis is the optical axis. The x-axis is vertical and the y-axis is horizontal. The whole calibration task is separated into three steps, as Figure 2 summarizes. The camera rotation between two images is first recovered using simple geometric constraints [6], as section 3.1 briefly introduces. We then use Eq. (3) to calibrate the camera's two rotational degrees of freedom and Eq. (4) to calibrate the third rotational degree of freedom and two translational degrees of freedom, as sections 3.2 and 3.3 describe, respectively. Finally, section 3.4 discusses how the calibrated parameters can be used for robot vision problems.

3.1 Recovering the camera rotation

To recover the camera's rotation between two frames, our strategy is first to infer the 3D structures from one perspective projection using the algorithms given in [6], and then to compute the relative rotation. We assume a rectangular corner to be the calibration reference, i.e., three mutually orthogonal half-lines intersecting at a point. We choose such a 3D structure because it can be easily found in indoor environments, and thus our calibration program does not need a special preset calibration object.

Using [6], the orientations of the three lines of a rectangular corner can be recovered up to a mirror image. If the correspondence of lines in two images can be established, we can compute the camera's rotation matrix. The computed rotation matrix, R_{cij}, is then used to estimate the camera's rotation axis, as section 3.2 describes. After that, the point correspondence of the corner in three images is used to calibrate the remaining parameters. Note that the computed R_{cij} is an orthonormal matrix if and only if the rigidity is preserved in the two frames. Since our structure recovery algorithm recovers only rectangular corners, the rigidity constraint is automatically satisfied.

3.2 Calibrating the rotation axis

The rotation transformation between the camera and the robot coordinate systems is represented by the yaw (ψ about the x-axis), the pitch (θ about the y-axis), and the roll (ϕ about the z-axis) angles. One may imagine that the camera first rotates ψ about the robot's x-axis; then it rotates θ about the new y-axis; and, finally, the camera rotates ϕ about the new z-axis. Thus, $P_c = R(\mathbf{z}, -\phi)\, R(\mathbf{y}, -\theta)\, R(\mathbf{x}, -\psi)\, P_g$, where P_g denotes the coordinates of a 3D point P in the robot's coordinate system, and P_c means the coordinates of the same point in the camera's system. We use $R(\mathbf{r}, \alpha)$ to denote the rotation of an angle α around an axis $\mathbf{r}$. From [7, pp. 30], if $\mathbf{r} = (r_x, r_y, r_z)$ is a unit vector, then the rotation matrix can be computed as

$$R(\mathbf{r}, \alpha) = \begin{bmatrix} r_x r_x \operatorname{vers}\alpha + \cos\alpha & r_y r_x \operatorname{vers}\alpha - r_z \sin\alpha & r_z r_x \operatorname{vers}\alpha + r_y \sin\alpha \\ r_x r_y \operatorname{vers}\alpha + r_z \sin\alpha & r_y r_y \operatorname{vers}\alpha + \cos\alpha & r_z r_y \operatorname{vers}\alpha - r_x \sin\alpha \\ r_x r_z \operatorname{vers}\alpha - r_y \sin\alpha & r_y r_z \operatorname{vers}\alpha + r_z \sin\alpha & r_z r_z \operatorname{vers}\alpha + \cos\alpha \end{bmatrix}, \quad (6)$$

where $\operatorname{vers}\alpha = 1 - \cos\alpha$. By the definition of a camera to robot transformation, we have

$$R_{cg} = R(\mathbf{x}, \psi)\, R(\mathbf{y}, \theta)\, R(\mathbf{z}, \phi). \quad (7)$$

From Eq. (3),

$$R_{cg}^t\, R_{gij}\, R_{cg} = R_{cij}. \quad (8)$$

But $R_{gij} = R(\mathbf{x}, \alpha)$, since we assume that the robot's rotational axis is fixed. Thus,

$$R(\mathbf{z}, -\phi)\, R(\mathbf{y}, -\theta)\, R(\mathbf{x}, -\psi)\, R(\mathbf{x}, \alpha)\, R(\mathbf{x}, \psi)\, R(\mathbf{y}, \theta)\, R(\mathbf{z}, \phi) = R_{cij}. \quad (9)$$

In general, matrix multiplication is not commutative but exceptional cases exist, such as $R(\mathbf{x}, \alpha)$ and $R(\mathbf{x}, \psi)$, because they have the same form. Thus, $R(\mathbf{x}, -\psi)$ and $R(\mathbf{x}, \psi)$ can cancel each other. We then have

$$R(\mathbf{z}, -\phi)\, R(\mathbf{y}, -\theta)\, R(\mathbf{x}, \alpha)\, R(\mathbf{y}, \theta)\, R(\mathbf{z}, \phi) = R_{cij}. \quad (10)$$

Comparing Eq. (10) with the derivation of Eq. (6) [5, pp. 21], one can see immediately that Eq. (10) indeed has the same physical meaning. That is, under the yaw-pitch-roll representation of the camera's rotation, the pitch and roll rotations transform the robot's rotation axis (x-axis) into the camera's rotation axis. Since both the pitch and roll angles are constant, as is the robot's rotational axis, the camera's rotation axis is thus constant, and the camera's rotation angle is equal to the robot's rotation angle. Note that the decomposition of the calibration task is only possible in the yaw-pitch-roll system. For example, if we change the order of rotation from yaw-pitch-roll to roll-pitch-yaw, then the computation of ϕ and θ requires solving ψ, which in turn requires estimating the translation at the same time.

Since the camera's rotation axis is fixed, we may estimate this axis instead of the two angles θ and ϕ. The rotation axis is computed using the fact that it is the eigen-vector of the rotation matrix R_{cij} with the eigen-value 1, using the algorithm taken from [9 and 7, pp. 30]. Note that the axis is undefined when $\alpha = 0$. But, since we have full control of the robot's motion, we can thus exclude the pure translation case. In the noiseless case, the rotation angle computed from R_{cij} should be equal to the robot's rotation angle α. Also note that to compute the rotation axis, we do not need in fact to know the robot's rotation angles. Thus, we can use the rotation angles computed from R_{cij} for other purposes:

(1) If the mobile robot's motion system has not been calibrated, the computed rotation angle can indeed be used to calibrate the mobile robot's rotation motion.

(2) The computed rotation angle, when compared with the robot's rotation angle, can be used as an estimate of the error in the current measurement from the image data. The information is needed to assign the weights in the Recursive Least Square algorithm.

(3) The computed rotation angle can be used to verify the recovered 3D structures. The application is specific to our algorithm. The problem arises because the inverse perspective geometry constraints we use can only recover the unknown 3D structure up to the mirrored images. Presumably, the correct structure of the two should result in a computed rotation angle closer to the robot's true rotation angle.

If the vision system has a better accuracy than the mechanical system, then we recommend using strategy (1). Otherwise, strategies (2) and (3) should be used (as in our current implementation). With a pair of images, we first recover the R_{cij} and then extract its rotation axis. The computed rotation axis is then used to sequentially update the estimate of the rotation axis using the Recursive Least Square algorithm. We treat each component of the axis as a scalar. Let x_i denote the i-th measurement of the quantity x, and let r_i be the variance of the error in x_i. Let x_{i-1} denote the estimate of x based on all previous measurements up to the (i-1)-th stage, and let r_{i-1} be the corresponding variance. Then the new data can be fused with the current estimate by

$$\mathit{x}_i = (\mathit{r}_{i-1} * x_i + r_i * \mathit{x}_{i-1}) / (r_i + \mathit{r}_{i-1}), \text{ and} \quad (11)$$

$$\mathit{r}_i = r_i * \mathit{r}_{i-1} / (r_i + \mathit{r}_{i-1}). \quad (12)$$

After updating the three components separately, we normalize the resulting vector to unity. x_0 is initialized by a random unit vector and r_0 is initialized as a large number.

In Eqs. (11) and (12), r_i is used as a weighting factor in the fusion of measurements. The error in the measurement is really generated by the noise in the raw image data. Therefore, theoretically r_i can be derived from given 2D image parameters and imaging noise statistics. However, in reality, the computed rotation axis is related to the original image data via very complicated functions. Thus, it is almost impossible to derive analytically the noise variance. Instead, we use the absolute difference in the computed rotation angle and the robot's rotation angle as an estimate of error. Such a replacement is justified by our experimental data presented in section 4. The weighting factors are important for the robustness of the estimation algorithm. Note that in our algorithm, the 3D structure is not assumed to be known *a priori*, but is instead recovered by using geometric constraints introduced in section 3.1. When the imaging noise is large and the 2D projection is near a singular configuration (such as the "T" structure), the recovery algorithm may choose the wrong structure and thus make a gross error. In such cases, the error in the computed rotation angle should also be large, and the wrong estimate will thus be assigned very little weight.

3.3 Calibrating the translation vector and the pan angle

Since we assume a rectangular corner as our calibration reference, we use the point correspondence of the corner's center to calibrate the remaining degrees of freedom. The center point can be reliably extracted by intersecting the three lines using least squares. In fact, one point correspondence in three images (four equations) is enough to recover the remaining three unknowns, but we use more images to suppress the noise effect. The most straightforward way to compute the unknowns is to solve the nonlinear equations by brute force. However, in our work we derive closed-form solutions by using algebraic manipulation. Appendix A provides the lengthy derivation. Basically, we first find the close-form expression for the roll angle ψ on known quantities such as the yaw and the pitch angles, and the robot's motion parameters. We then derive similar expressions for the two translation parameters, $T_{cg,y}$ and $T_{cg,z}$, independent of ψ.

An important issue in numerical algorithms is the noise sensitivity. Unfortunately, from our simulation we found that the second stage computation may have problems with outliers, as Figure 3 shows. Because of the decomposition of the calibration task, the second stage computation depends on the results from the first stage via nonlinear (mostly sinusoidal) functions. Therefore, the errors in the first stage parameters, even if they can be modeled as a zero-mean white Gaussian noise, may generate non-Gaussian, correlated, or even biased noise for the second stage parameters. The problem cannot be solved by the RLS approach because the second stage errors may sometimes have an unbound variance. Haralick et al. [5] study a similar problem in the computation of structure from motion, and suggest robust estimation techniques as a solution. Our current work uses the order statistics approach to attain robustness. We use moving windows of size eleven (initialized by zeros) to store measurements of translation and yaw (pan) angle computed from every three images. The eleven numbers in each moving window are then sorted according to their magnitudes and the average of the middle seven is used as the current estimate of the parameter. In other words, we always rely on the eleven most recent measurements, and we always treat the four measurements most far away from the median in each window as outliers. The algorithm is simple and works well against noise, as section 4 discusses.

3.4 The navigation problem

The ultimate goal of our calibration task is to provide the transformation between camera motion and robot motion. Obviously, Eq. (8) can be used to compute the camera's rotation from the robot's rotation. Even better, since we estimated the camera's fixed rotation axis and we also know that the camera's rotation angle should be equal to the robot's, R_{cij} can be easily computed by Eq. (6). The camera's translation can be computed by rewriting Eq. (4) as

$$T_{cij} = R_{cg}^{-1} (R_{gij} - I)\, T_{cg} + R_{cg}^{-1}\, T_{gij}. \qquad (4')$$

Note that $(R_{gij} - I)$ always annihilates the x-component of T_{cg}. Therefore, if we are only interested in computing T_{cij} from given robot motion R_{gij} and T_{gij}, we do not need to know $T_{cg,x}$.

4. Experimental results

The proposed algorithm is tested with both simulation and real data. We first discuss simulation results in section 4.1 and then describe our experimental setup in section 4.2.

4.1 Simulation

In the simulation case, to generate synthetic images we consider three sets of configurations with different camera, motion, and 3D structure parameters. A sequence of images is generated by projecting the rectangular corner via the synthetic camera mounted on the mobile robot, which moves with the specified sequence of motion commands. Figure 4 shows the three example images. Examples a and c are each a sequence of twenty projections of a rectangular corner superimposed on one image, while example b has thirty projections. Table 1 shows the three camera parameters. All the translations are in feet, and the rotations are all in radians. Note that the three cameras have compatible translations and rotation angles. We evaluate the proposed algorithm by adding zero-mean Gaussian noise, characterized by its standard deviation, to each pixel on the image lines. The line parameters are then computed by the least square fitting. Thus, the length of the line segment is very important for the estimate's accuracy. The average lengths are about 107, 72, and 87 pixels in the first, second, and third examples, respectively. We also add Gaussian noise, characterized by the SNR (signal to noise ratio) to the motion parameters, including one rotation (pan), two translations (y and z), and the height of the camera (to model the surface's roughness). The three motion components (in ft. and rad., respectively) have compatible magnitudes (about 0.35 in average for all examples). The 3D structure and motion parameters are generated randomly first and then modified manually to meet the conflicting requirements: 1) the motion parameters should be as large as possible; 2) the 2D line segments should be as long as possible, and 3) all lines must stay inside the camera's views as much as possible. Indeed, view selection is crucial for the calibration task and is an interesting problem on its own. We are currently investigating an automatic way to generate a sequence of best views for calibration.

To evaluate the calibration results, we define the percentage error in the rotation axis as the norm of the difference vector between the true and the estimated axes, which are both unit vectors. The percentage error in the pan angle is defined as the absolute difference of the true and estimated rotation angles divided by the true magnitude. The percentage error in the translation (y and z components only) is defined as the distance between the true and the estimated 2D translation vectors divided by the length of the true vector.

The proposed algorithm is tested by the three example images under six imaging and six motion noise conditions. The only heuristic parameters used are for assigning the weights in the RLS computation. As Figure 5 demonstrates, we can use the approximate:

$$\text{error\_in\_axis} \approx 5.0 * \text{error\_in\_angle} + 0.01. \qquad (13)$$

The same parameters are used in all performed tests. Tables 2a, 2b, and 2c present the statistics of the calibration results for the rotation axis, the pan angle, and the translation, respectively, while Figure 6 shows an example of the convergence of the estimations. To obtain the statistics, we generate ten realizations under each condition by changing the initial guess of the rotation axis and the seed used by the pseudo random number generator. Each entry in the table thus represents an average of thirty samples. From the table, we can find the range of noise conditions under which our calibration algorithm can give satisfactory results. In our opinion, our algorithm performs well with an imaging noise standard deviation of less than four pixels and a motion SNR greater than 24 dB. It is worth pointing out that the rotation axis estimation is still acceptable, even under severe motion noise, due to the fact that in our algorithm, the rotation axis estimate is computed independently of the robot's motion parameters.

4.2 Real world implementation

We set up a robot vision system on the TRC LABMATE mobile robot. The CCD camera mounted on the mobile robot is first calibrated [10] to obtain the intrinsic parameters. Due to the slippery floor, the robot has a rather large rotation motion error (or bias), sometimes as large as 20%. Therefore, we are only able to obtain results for the calibration of the rotation axis, since that part does not depend on the robot's accuracy.

We test the rotation axis estimation part of our algorithm using a sequence of ten images. Fig. 7 shows a typical image. Fig. 8 contains the detected line segments, and Fig. 9 is the rectangular corner extracted by grouping lines manually. The following is a list of the rotation axis estimated by our algorithm for every two images and also the final result.

stage-1: (0.927535, 0.120115, -0.353909)
stage-2: (0.962302, 0.113994, -0.246940)
stage-3: (0.997704, 0.026500, -0.062329)
stage-4: (0.933305, -0.056756, -0.354570)
stage-5: (0.948445, -0.054887, -0.312154)
stage-6: (0.682508, -0.306450, -0.663529)
stage-7: (0.964626, - 0.39264, -0.260684)
stage-8: (0.966549, -0.041660, -0.253078)
stage-9: (0.986979, -0.071003, -0.144330)
final: (0.936364, -0.050037, -0.347445).

Note that due to poor image quality, the 6th stage computation provides a much less reliable estimate than others. But thanks to our robust RLS algorithm, the effect of this bad estimate is minimal.

Since we do not have the ground truth of our robot vision setup, we are unable to verify the calibration results accurately. By direct measurements, using a protractor, we roughly know that the pitch angle is about -20° (note that the camera is looking down) and that the roll angle is almost zero. Thus, the rotational axis, represented in the camera's coordinate system, should be approximately (0.939693, 0, -0.34202), which is very close to the calibration result. We expect to have a more rigorous verification of our algorithm.

5. Conclusions

In summary, this paper studies the problem of estimating the fixed transformation between the camera's and the mobile robot's coordinate systems. The information is needed to relate the robot motion to the camera motion. Important applications of the proposed algorithm may include such tasks as active vision, mobile robot navigation, path planning, obstacle avoidance, etc.

We show that even though the mobile robot has only one rotational degree of freedom, with the proper representation (yaw-pitch-roll) of the camera rotation, all unknowns in the camera/robot transformation can still be computed except $T_{cg,x}$ (height). We then derive the closed-form solutions for the yaw (pan) angle and the translation. We argue that for some robot navigation and active vision problems, the height information ($T_{cg,x}$) may not be needed, and we provide a formula for transforming the uncertainty of parameters that explicitly considers the uncertainty in the robot motion.

The proposed computation algorithm assumes that the camera's intrinsic parameters are known and that the projection of a rectangular corner (three line segments) is extracted. The orientations of the 3D half-lines in one image are then recovered. With a pair of such images, the camera rotation R_{cij} is computed, and the camera's rotation axis can be recovered. With three images, the remaining parameters can all be computed with one point correspondence. Figure 2 summarizes the whole algorithm. To obtain robustness, the weighted RLS algorithm is used to fuse the new measurement of the rotation axis with the previous estimate, and the rank-order filter prevents gross errors in the computation of the pan angle and the translation. The proposed algorithm does not require a pre-designed calibration pattern, and the results from both the simulation and the real data show that the algorithm performs well under realistic imaging and motion noise conditions.

References

[1] R. A. Brooks, A. M. Flynn, and T. Marill, "Self calibration of motion and stereo vision for mobile robots," in *Proc. DARPA Image Under. Work.*, pp. 267 - 276, 1987.
[2] H. F. Durrant-Whyte, "Uncertain geometry in robotics," in *IEEE Trans. Robotics Automation*, vol. 4, no. 1, pp. 23 - 31, Feb. 1988.
[3] O. D. Faugeras and G. Toscani, "Camera calibration for 3D computer vision," in *Proc. Int. Work. Machine Vision Machine Intelligence*, Japan, Feb. 1987.
[4] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill: New York, 1987.
[5] R. M. Haralick et al, "Pose estimation from corresponding point data," in *IEEE Trans. Sys. Man Cyb.*, vol. 19, no. 6, pp. 1426 - 1445, Nov./Dec. 1988.
[6] K.-I. Kanatani, "Constraints on length and angle," *CVGIP* **41**, pp. 28 - 42, 1988.
[7] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press: Cambridge, MA, 1982.
[8] P. Puget and T. Skordas, "An optimal solution for mobile camera calibration," in *Proc. IEEE Int. Conf. Robotics Automat.*, pp. 34 - 39, May, 1990.
[9] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX = XB," *IEEE Trans. Robotics Automat.*, vol. 5, no. 1, pp. 16 - 29, Feb., 1989.
[10] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Trans. Robotics Automat.*, vol. RA-3, no. 4, pp. 323 -344, Aug., 1987.
[11] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Trans. Robotics Automat.*, vol. 5, no. 3, pp. 345 -358, June, 1989.
[12] Z. Zhang and O. D. Faugeras, "Calibration of a mobile robot with application to visual navigation," in *Proc. IEEE Work. Visual Motion*, pp. 306 - 313, Mar., 1989.

Appendix A. Computing the Yaw and the Translation

Assume that a point has been located in three images and that it has coordinates (p_x, p_y), (p_x', p_y'), and (p_x'', p_y'') on the image planes. From inverse perspective geometry, the actual 3D point must have coordinates in the three camera frames as $P_c = sp = s\,(p_x, p_y, f)^t$, $P_c' = s'p' = s'(p_x', p_y', f)^t$, and $P_c'' = s''(p_x'', p_y'', f)^t$, where s, s', and s'' are the three unknown structure parameters and f is the focal length.

From the definition of the camera-to-robot transformation, the 3D point must have coordinates in the robot frame as

$$P_g = R_{cg}\, P_c + T_{cg}, \qquad \text{(A.1)}$$

while P_g' and P_g'' also have a similar expression. Assume that the mobile robot has the movement R_g and T_g between the first and second frames and R_g' and T_g' between the second and third frames. From the definition of the robot motion transformation, we have

$$P_g' = R_g\, P_g + T_g = R_g\, R_{cg}\, P_c + R_g\, T_{cg} + T_g = R_{cg}\, P_c' + T_{cg}. \qquad \text{(A.2)}$$

Relating the unknowns to the measurements from images, we have

$$R_g\, R_{cg}\, ps + R_g\, T_{cg} + T_g = R_{cg}\, p's' + T_{cg}. \qquad \text{(A.3)}$$

Now we invoke our assumption about the robot rotation motion and our model of the camera rotation, i.e.,

$$R_g = R(\mathbf{x}, \alpha),\ R_{cg} = R(\mathbf{x}, \psi)\, R(\mathbf{y}, \theta)\, R(\mathbf{z}, \phi) = R(\mathbf{x}, \psi)\, R(\theta, \phi),$$

where $R(\theta, \phi) = R(\mathbf{x}, \psi)\, R(\mathbf{y}, \theta)$. (A.4)

Hence, Eq. (A.3) can be rewritten as

$$R(\mathbf{x}, \alpha)\, R(\mathbf{x}, \psi)\, R(\theta, \phi)\, ps + R(\mathbf{x}, \alpha)\, T_{cg} + T_g$$
$$= R(\mathbf{x}, \psi)\, R(\theta, \phi)\, p's' + T_{cg}. \qquad \text{(A.5)}$$

Since $R(\theta, \phi)$ has already been calibrated in the first stage, the known data vector can be transformed as

$$\mathit{p} = R(\theta, \phi)\, p,\ \mathit{p}' = R(\theta, \phi)\, p'. \qquad \text{(A.6)}$$

Eq. (A.5) can thus be rewritten as

$$R(\mathbf{x}, \alpha)\ R(\mathbf{x}, \psi)\, \mathit{ps} + R(\mathbf{x}, \alpha)\ T_{cg} + T_g = R(\mathbf{x}, \psi)\, \mathit{p}'s' + T_{cg}. \qquad \text{(A.7a)}$$

Notice that we can exchange the order of the multiplication of $R(\mathbf{x}, \alpha)$ and $R(\mathbf{x}, \psi)$, i.e.,

$$R(\mathbf{x}, \psi)\ R(\mathbf{x}, \alpha)\, \mathit{ps} + R(\mathbf{x}, \alpha)\ T_{cg} + T_g = R(\mathbf{x}, \psi)\, \mathit{p}'s' + T_{cg}. \qquad \text{(A.7b)}$$

Since the robot's movement is known, the measurement vector can be transformed again as

$$\underline{p} = R(\mathbf{x}, \alpha)\, \mathit{p}. \qquad \text{(A.8)}$$

Eq. (A.7b) can now be rewritten as

$$R(x,\psi)\,\underline{p}s + R(x,\alpha)\,T_{cg} + T_g = R(x,\psi)\,p's' + T_{cg}, \qquad \text{(A.9a)}$$

or more compactly, as

$$R(x,\psi)\,(p's' - \underline{p}s) + (I - R(x,\alpha))\,T_{cg} = T_g. \qquad \text{(A.9b)}$$

A similar equation can also be obtained for the second and third frames.

Note that since we assume that the mobile robot can only execute planar motion, $T_{g,x}$ is always zero and the robot can only rotate around its x-axis. Thus we cannot solve for $T_{cg,x}$. On the other hand, the relation between the two structure parameters, s and s', can easily be found as

$$s = (p'_x/\underline{p}_x)\,s' = k \bullet s'. \qquad \text{(A.10)}$$

Thus Eq. (A.9b) can be reduced to

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} p'_y - k\underline{p}_y \\ p'_z - k\underline{p}_z \end{bmatrix}s' + \begin{bmatrix} 1-\cos\alpha & \sin\alpha \\ -\sin\alpha & 1-\cos\alpha \end{bmatrix}\begin{bmatrix} T_{cg,y} \\ T_{cg,z} \end{bmatrix} = \begin{bmatrix} T_{g,y} \\ T_{g,z} \end{bmatrix}. \qquad \text{(A.11a)}$$

Similarly, for the second and third frames,

$$s'' = (\underline{p}'_x/p''_x)\,s' = k'' \bullet s'. \qquad \text{(A.12)}$$

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} k''p''_y - \underline{p}'_y \\ k''p''_z - \underline{p}'_z \end{bmatrix}s' + \begin{bmatrix} 1-\cos\alpha' & \sin\alpha' \\ -\sin\alpha' & 1-\cos\alpha' \end{bmatrix}\begin{bmatrix} T_{cg,y} \\ T_{cg,z} \end{bmatrix} = \begin{bmatrix} T'_{g,y} \\ T'_{g,z} \end{bmatrix}. \qquad \text{(A.13a)}$$

We can rewrite Eqs. (A.11a) and (A.13a) more compactly as

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}s' + \begin{bmatrix} c & d \\ -d & c \end{bmatrix}\begin{bmatrix} T_{cg,y} \\ T_{cg,z} \end{bmatrix} = \begin{bmatrix} T_{g,y} \\ T_{g,z} \end{bmatrix}. \qquad \text{(A.11b)}$$

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} a' \\ b' \end{bmatrix}s' + \begin{bmatrix} c' & d' \\ -d' & c' \end{bmatrix}\begin{bmatrix} T_{cg,y} \\ T_{cg,z} \end{bmatrix} = \begin{bmatrix} T'_{g,y} \\ T'_{g,z} \end{bmatrix}. \qquad \text{(A.13b)}$$

Now we want to solve for the unknown angle ψ first. Therefore, we need to cancel the unknown translation in the two equation systems. We first invert their coefficient matrices:

$$\begin{bmatrix} \chi & -\delta \\ \delta & \chi \end{bmatrix}\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}s' + \begin{bmatrix} T_{cg,y} \\ T_{cg,z} \end{bmatrix} = \begin{bmatrix} \chi & -\delta \\ \delta & \chi \end{bmatrix}\begin{bmatrix} T_{g,y} \\ T_{g,z} \end{bmatrix},$$

$$\text{where} \begin{bmatrix} \chi & -\delta \\ \delta & \chi \end{bmatrix} = \begin{bmatrix} c & d \\ -d & c \end{bmatrix}^{-1}. \qquad \text{(A.14)}$$

Eq. (A.11b) finally becomes

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} A \\ B \end{bmatrix}s' + \begin{bmatrix} T_{cg,y} \\ T_{cg,z} \end{bmatrix} = \begin{bmatrix} \mathcal{T}_y \\ \mathcal{T}_z \end{bmatrix},$$

$$\text{where} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \chi & -\delta \\ \delta & \chi \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}, \text{ and } \begin{bmatrix} \mathcal{T}_y \\ \mathcal{T}_z \end{bmatrix} = \begin{bmatrix} \chi & -\delta \\ \delta & \chi \end{bmatrix}\begin{bmatrix} T_{g,y} \\ T_{g,z} \end{bmatrix}. \qquad \text{(A.15)}$$

Similarly, for Eq. (A.13b),

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} A' \\ B' \end{bmatrix}s' + \begin{bmatrix} T_{cg,y} \\ T_{cg,z} \end{bmatrix} = \begin{bmatrix} \mathcal{T}'_y \\ \mathcal{T}'_z \end{bmatrix}. \qquad \text{(A.16)}$$

Now we subtract Eq. (A.16) from (A.15):

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} A - A' \\ B - B' \end{bmatrix}s' = \begin{bmatrix} \mathcal{T}_y - \mathcal{T}'_y \\ \mathcal{T}_z - \mathcal{T}'_z \end{bmatrix}, \qquad \text{(A.17a)}$$

or, more compactly,

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}s' = \begin{bmatrix} \mathbf{T}_y \\ \mathbf{T}_z \end{bmatrix}. \qquad \text{(A.17b)}$$

Finally, the angle ψ can be easily computed by

$$\psi = \tan^{-1}((\mathbf{T}_z\mathbf{A} - \mathbf{T}_y\mathbf{B})\,/\,(\mathbf{T}_y\mathbf{A} + \mathbf{T}_z\mathbf{B})). \qquad \text{(A.18)}$$

To compute the unknown translation, an easy way is to first solve for the structure parameter s' using Eq. (A.17b), and then to substitute ψ and s' into Eq.(A.15) or (A.16). However, from simulation, we found that this algorithm is sensitive to noise, since the computed translation depends on both ψ and s'. Instead, we try to solve for $T_{cg,y}$ and $T_{cg,z}$ directly from Eqs. (A.15) and (A.16). First we write down both equation systems explicitly as

$$(A\cos\psi - B\sin\psi)\,s' + T_{cg,y} = \mathcal{T}_y. \qquad \text{(A.19)}$$

$$(A\sin\psi + B\cos\psi)\,s' + T_{cg,z} = \mathcal{T}_z. \qquad \text{(A.20)}$$

$$(A'\cos\psi - B'\sin\psi)\,s' + T_{cg,y} = \mathcal{T}'_y. \qquad \text{(A.21)}$$

$$(A'\sin\psi + B'\cos\psi)\,s' + T_{cg,z} = \mathcal{T}'_z. \qquad \text{(A.22)}$$

Now we take A • (A.19) + B • (A.20), and also A' • (A.21) + B' • (A.22). We shall have

$$(A^2 + B^2)\,s'\cos\psi + (AT_{cg,y} + BT_{cg,z}) = A\mathcal{T}_y + B\mathcal{T}_z. \qquad \text{(A.23)}$$

$$(A'^2 + B'^2)\,s'\cos\psi + (A'T_{cg,y} + B'T_{cg,z}) = A'\mathcal{T}'_y + B'\mathcal{T}'_z. \qquad \text{(A.24)}$$

We then take $(A'^2 + B'^2)$ • (A.23) - $(A^2 + B^2)$ • (A.24) to form a new equation:

$$(A'^2 + B'^2)\,(AT_{cg,y} + BT_{cg,z}) - (A^2 + B^2)\,(A'T_{cg,y} + B'T_{cg,z})$$
$$= (A'^2 + B'^2)\,(A\mathcal{T}_y + B\mathcal{T}_z) - (A^2 + B^2)\,(A'\mathcal{T}'_y + B'\mathcal{T}'_z). \qquad \text{(A.25)}$$

Similarly, we can take A' • (A.19) - A • (A.21), and also B' • (A.20) - B • (A.22). Thus,

$$(AB' - A'B)\,s'\sin\psi + (A' - A)T_{cg,y} = A'\mathcal{T}_y - A\mathcal{T}'_y. \qquad \text{(A.26)}$$

$$(AB' - A'B)\,s'\sin\psi + (B' - B)T_{cg,z} = B'\mathcal{T}_z - B\mathcal{T}'_z. \qquad \text{(A.27)}$$

Simply taking (A.26) - (A.27), we have

$$(A' - A)T_{cg,y} - (B' - B)T_{cg,z} = A'\mathcal{T}_y - A\mathcal{T}'_y - B'\mathcal{T}_z + B\mathcal{T}'_z. \qquad \text{(A.28)}$$

Since A, B, A', B', $\mathcal{T}_y$, $\mathcal{T}'_y$, $\mathcal{T}_z$, and $\mathcal{T}'_z$ are all known, both $T_{cg,y}$ and $T_{cg,z}$ can be solved using Eqs. (A.25) and (A.28).

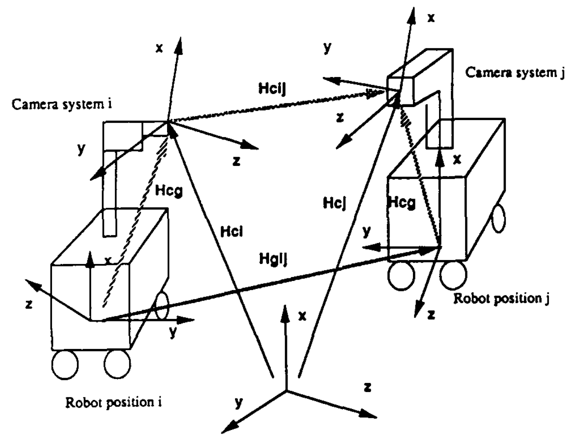

Figure 1. The definition of the hand/eye calibration problem.

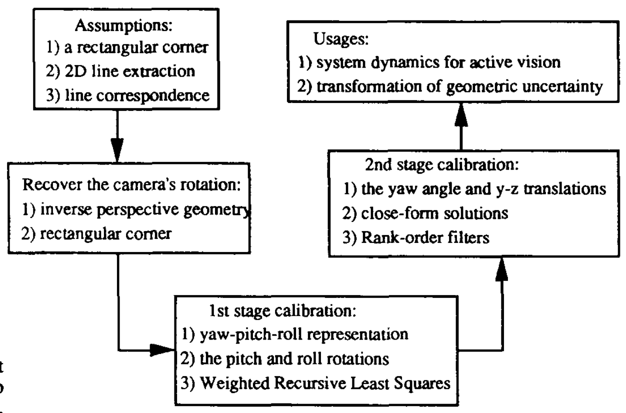
Assumptions:
1) a rectangular corner
2) 2D line extraction
3) line correspondence
Usages:
1) system dynamics for active vision
2) transformation of geometric uncertainty
Recover the camera's rotation:
1) inverse perspective geometry
2) rectangular corner
2nd stage calibration:
1) the yaw angle and y-z translations
2) close-form solutions
3) Rank-order filters
1st stage calibration:
1) yaw-pitch-roll representation
2) the pitch and roll rotations
3) Weighted Recursive Least Squares

Figure 2. A summary of the whole algorithm.

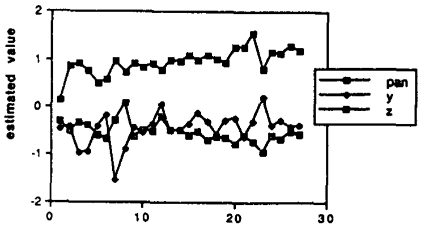

Figure 3. Outliers in the second stage computation. (Imaging noise = 3 pxl, motion SNR = 29.5, camera-1 in use)

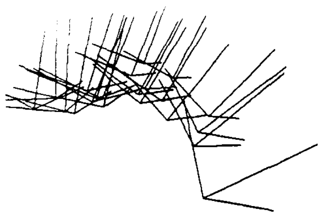

Figure 4a. The 1st example, with 20 projections of the rectangular corner, average length of the line segments = 87 pixels.

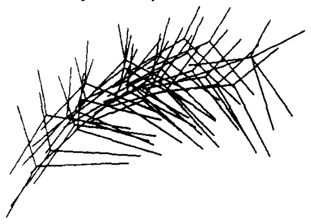

Figure 4b. The 2nd example, with 30 projections of the rectangular corner, average length of the line segments = 72 pixels.

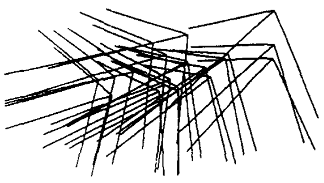

Figure 4c. The 3rd example, with 20 projections of the rectangular corner, average length of the line segments = 107 pixels.

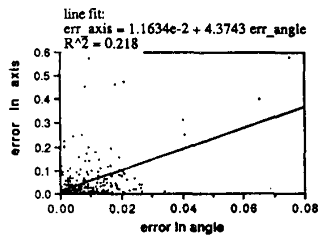

Figure 5. Error in angle vs. error in axis

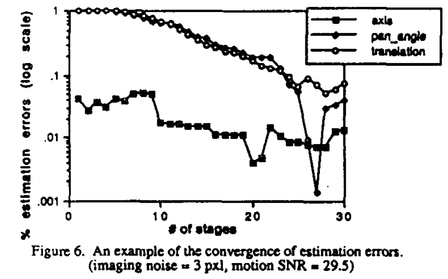

Figure 6. An example of the convergence of estimation errors. (imaging noise = 3 pxl, motion SNR = 29.5)

	rotation (yaw-pitch -roll) in raidan	translation (x-y-z) in foot	image center (x-y) in pixel	focal length (x-y) in pixel
camera-1	(-0.6, 1.0, -0.2)	(1.0, -0.4, 1.1)	(256, 240)	(400, 400)
camera-2	(-0.3, -0.9, -0.2)	(7.0, -1.3, 0.6)	(256, 240)	(400, 400)
camera-3	(-0.3, -0.8, 0.0)	(7.0, -0.2, 1.0)	(256, 240)	(400, 400)

Table 1. The three camera parameters

imaging noise (pixels) / motion noise (SNR)	0	1	2	3	4	5
9.5	0.029505	0.029434	0.033974	0.037932	0.044376	0.052328
23.5	0.005225	0.006641	0.011604	0.018610	0.024588	0.033431
29.5	0.003820	0.005347	0.009767	0.015024	0.020713	0.027717
35.5	0.002530	0.004567	0.008106	0.012581	0.020555	0.028995
43.5	0.001846	0.004177	0.007538	0.011960	0.016742	0.023633
49.5	0.001576	0.004019	0.007609	0.012516	0.017695	0.023113

Table 2a. The average % error in the axis estimation.

imaging noise (pixels) / motion noise (SNR)	0	1	2	3	4	5
9.5	0.806392	0.778933	0.718718	0.718923	0.685196	0.646715
23.5	0.170195	0.180625	0.174323	0.170920	0.181849	0.203529
29.5	0.076447	0.072184	0.077709	0.129327	0.117307	0.183035
35.5	0.043591	0.047928	0.054568	0.100253	0.140063	0.164977
43.5	0.014245	0.023182	0.045140	0.066403	0.079675	0.094211
49.5	0.010273	0.025736	0.039051	0.057343	0.072303	0.096638

Table 2b. The average % error in the pan angle estimation.

imaging noise (pixels) / motion noise (SNR)	0	1	2	3	4	5
9.5	0.633263	0.657006	0.662206	0.620064	0.627336	0.639910
23.5	0.135719	0.143059	0.156563	0.174638	0.207303	0.257427
29.5	0.064748	0.072394	0.089599	0.112575	0.154948	0.196606
35.5	0.035596	0.045692	0.065874	0.097211	0.153413	0.200651
43.5	0.023951	0.040255	0.065576	0.090017	0.115935	0.152313
49.5	0.022005	0.039371	0.063718	0.097108	0.125028	0.163137

Table 2c. The average % error in the translation estimation.

Table 2. The simulation results

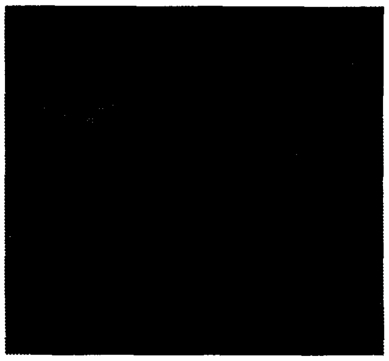

Figure 7. An example of the images used for calibration.

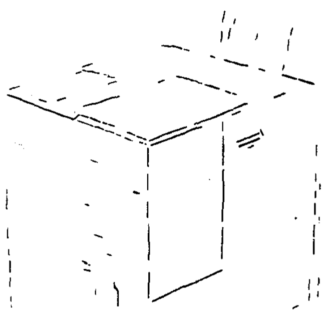

Figure 8. Line segments detected.

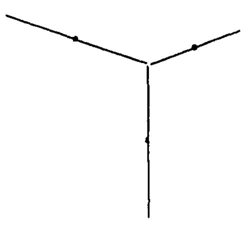

Figure 9. The rectangular corner extracted.